Beyond Scaling: Predicting Patent Approval with Domain-specific Fine-grained Claim Dependency Graph

Xiaochen Kev Gao[1*], **Feng Yao**[1*], **Kewen Zhao**[2], **Beilei He**[3], **Animesh Kumar**[1], **Vish Krishnan**[1], **Jingbo Shang**[1]

[1] University of California San Diego,
[2] Carnegie Mellon University, [3] University of Pennsylvania
{xig034,fengyao,ank028,vkrishnan,jshang}@ucsd.edu
{kewenz}@cs.cmu.edu, {beilei}@seas.upenn.edu

Abstract

Model scaling is becoming the default choice for many language tasks due to the success of large language models (LLMs). However, it can fall short in specific scenarios where simple customized methods excel. In this paper, we delve into the patent approval prediction task and unveil that simple domain-specific graph methods outperform enlarging the model, using the intrinsic dependencies within the patent data. Specifically, we first extend the embedding-based state-of-the-art (SOTA) by scaling up its backbone model with various sizes of open-source LLMs, then explore prompt-based methods to harness proprietary LLMs' potential, but find the best results close to random guessing, underlining the ineffectiveness of model scaling-up. Hence, we propose a novel Fine-grained cLAim depeNdency (FLAN) Graph through meticulous patent data analyses, capturing the inherent dependencies across segments of the patent text. As it is model-agnostic, we apply cost-effective graph models to our FLAN Graph to obtain representations for approval prediction. Extensive experiments and detailed analyses prove that incorporating FLAN Graph via various graph models consistently outperforms all LLM baselines significantly. We hope that our observations and analyses in this paper can bring more attention to this challenging task and prompt further research into the limitations of LLMs. Our source code and dataset can be obtained from https://github.com/ShangDataLab/FLAN-Graph.

1 Introduction

Scaling up language models has demonstrated predictable improvement and unprecedented abilities in many language tasks (Chung et al., 2022; Wei et al., 2022a; Zhao et al., 2023). However, emerging evidence shows that simply scaling up backbone models to large language models (LLMs) may not guarantee success (Peng et al., 2023; Hou et al., 2023; Wang et al., 2023). In addition, scaling up models imposes demanding computational costs that prevent it from being widely adopted for real-world applications. Such limitations necessitate cost-effective methods beyond scaling, especially for domain-specific tasks that have distinct traits.

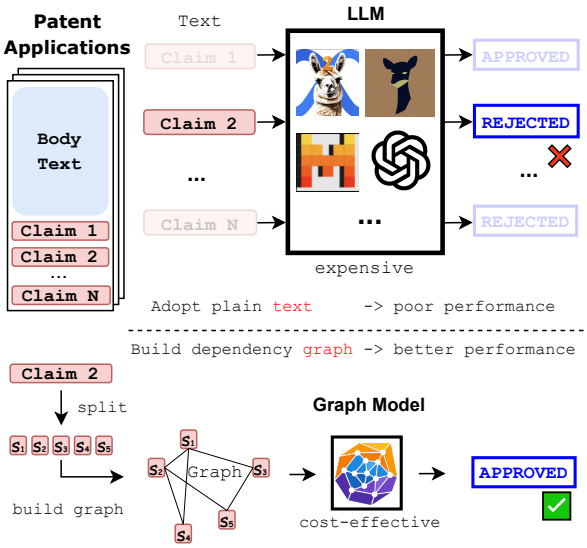


Figure 1: An illustration for the patent approval prediction task approached by LLMs and graph models, where each node of the graph is an informative segment decomposed from the original claim text.

In this paper, we look into the task of patent approval prediction, a challenging yet straightforward classification task that scaling struggles to address, and explore customized cost-effective solutions. As shown in Figure 1, the objective is to determine if each claim in a patent application will be approved or rejected by the U.S. government patent office (USPTO). It is vital for intellectual property (IP) protection, taking up to 40% of the U.S. GDP and over 30% of employment. Due to the demanding requirements for knowledge in both technology and law, patent examination is conducted manually,

*Equal contribution. Listing order is random.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 5218–5234
August 11-16, 2024 ©2024 Association for Computational Linguistics

leading to potential inconsistent outcomes across patent examiners (O'Neill, 2018; USPTO, 2016). Such inconsistency underscores the necessity for objective and automated computational support.

The state-of-the-art (SOTA) of this task is based on BERT embedding (Devlin et al., 2019) augmented with handcrafted features (Gao et al., 2022). An intuitive idea is to replace its backbone model with modern LLMs. To this end, we employ LLaMA2 (Touvron et al., 2023), Mistral (Jiang et al., 2023), Vicuna (Chiang et al., 2023) in various sizes (7&13&70B) and apply both LoRA (Hu et al., 2022) and full fine-tuning. Surprisingly, they do not live up to expectations, performing on par or worse than BERT. To exploit LLMs' emergent abilities, we utilize prompt-based methods tailored to these open-source LLMs, as well as closed-source GPT-3.5 (OpenAI, 2022) and GPT-4 (OpenAI, 2023), but the results are still unsatisfying.

The shattered hope in LLMs motivates us to dive into patent data analyses, which leads us to the standardized writing of claims and the dependencies nature among them. As depicted in Figure 2, *Claim 1* compromises multiple sub-components, which are then referenced in subsequent claims. Such intricate inner-claim (between sub-components in *Claim 1*) and inter-claim dependencies (between *Claims 1&2* as well as *Claims 1&3*) have critical implications for the patent approval prediction task, as the patent examination is conducted on each claim and the rejection of one claim can result in the automatic rejection of its dependents.

Inspired by the observations and domain-specific knowledge acquired from painstakingly extensive data analyses, we propose Fine-grained cLAim depeNdency (FLAN) Graph for patent approval prediction, which represents each claim by a single graph that encapsulates both inner- and inter-claim dependencies. Specifically, as shown in Figure 1, we first design a novel algorithm to automatically construct the FLAN Graph at scale, where each node is an informative segment of the claim text. Then, the model-agnostic FLAN Graph is fed into a generic graph model for prediction. Examples of the FLAN graphs and the corresponding claims are shown in Appendix C. In the experiments, we adopt a variety of cost-effective graph models such as GCN (Chen et al., 2020), GAT (Velickovic et al., 2018), and TreeLSTM (Tai et al., 2015) to verify the effectiveness of our proposed FLAN Graph for patent approval prediction. All models with FLAN Graph applied outperform the previous SOTA, among which GraphSage (Hamilton et al., 2017) achieves the highest improvements of 7.4% in AUC and 7.8% in Macro-F1 scores, achieving absolute scores of 66.04 and 58.22, respectively.

Claim 1: A system for [...], *the system compromising*:
an authentication component configured to [...];
a tracking component configured to [...];
and a control component configured to:
receive authentication information [...];
receive location information from [...];
and deliver a message to the touchpoint authorizing [...].

Claim 2: The system of claim 1, where the control component is also configured [...].

Claim 3: The system of claim 1, where the authentication component includes [...].

Figure 2: A brief example of the typical patent claim writing style and hierarchical dependencies within claims from a real-world patent application.

To summarize, our contributions are two-fold: **(1)** We propose a novel algorithm to automatically construct the Fine-grained cLAim depeNdency (FLAN) Graph at scale that consistently improves the SOTA by a large margin. **(2)** We conduct comprehensive experiments and analyses of modern LLMs on patent approval prediction, which identify the limitations of LLMs and provide valuable references for developing LLM-based solutions in the future. The source code[1] and dataset[2] are publicly released to facilitate future research.

2 Problem Formulation

In this section, we formally introduce the definition of the patent approval prediction task and analyze the dataset we construct for experiments.

2.1 Task Definition

As illustrated in Figure 1, patent applications are initially submitted to the USPTO in the form of documents. The examination process, however, focuses on approving or rejecting each individual claim. Therefore, given a patent application $A_i = \{C_j^{(i)}\}_{j=1}^{n}$ containing n claims, the task of patent approval prediction is to determine whether each claim $C_j^{(i)}$ will be approved or rejected by the USPTO indicated by a binary label $y_j^{(i)} \in \{0, 1\}$.

In practice, patent claims are reviewed according to the legal section 35 U.S. Code § 102, where the core criterion is novelty-based, bringing in dis-

[1] https://github.com/ShangDataLab/FLAN-Graph
[2] https://huggingface.co/datasets/ShangDataLab-UCSD/PatentAP

	#Claim	#Application	Approval (%)
Train	1,485,693	87,883	81.36
Valid	278,215	16,955	83.41
Test	185,477	11,148	84.92

Table 1: Statistics of PATENTAP dataset. "Approval (%)" indicates the percentage of the approved claims.

tinct challenges below. (1) **Time-sensitive**. Unlike traditional text classification, novelty assessment depends on the application filing date, allowing opposite decisions for the same claim over time. (2) **Structure-dependent**. Many claims (e.g., *Claims 2&3* in Figure 2) are dependent on others within the same application, and such structure can influence novelty evaluation. (3) **Knowledge-intensive**. Evaluating novelty requires up-to-date knowledge of both technologies and patent law. (4) **Outcome-inconsitent**. Novelty examination outcomes are subject to preferences across patent officers, which may introduce inconsistencies in patent data.

2.2 Dataset Collection

We collect data of real-world patent applications from Gao et al. (2022) and filter out those outdated data before 2018. The data is initially merged and derived from the publicly available resources officially released on the USPTO websites[3].

Considering the real-world scenario, we utilize historical data for training and more recent data for evaluation. Specifically, we sort the applications based on filing dates and then split them into training, validation, and test sets. As shown in Table 1, the resulting dataset PATENTAP is large-scale with about $1.5M$ claims for training and $0.5M$ for evaluation. It is also highly imbalanced, with most claims being approved, further adding to the difficulty. The claims are relatively short and 92% have less than 128 tokens and the average length is 54. Each application has 17 claims on average.

3 Methodology

In this section, we delve into the details of our proposed Fine-grained cLAim depeNdency (FLAN) Graph. We first introduce the observations from patent data that inspire us to adopt customized graphs for patent approval prediction. Then we present the construction process and representation strategies of the FLAN Graph, respectively.

[3] https://ped.uspto.gov/peds/#/

3.1 Observations

In principle, patent claims are filed to seek legal protection for complex systems that usually compromise multiple (sub-)components. Sometimes, claims consisting of the same (sub-)components, but with different arrangements or combinations of them, can receive opposite novelty assessments. Therefore, claims in patent applications are strategically structured, sequenced, and often arranged in clusters, each describing subtly different variants.

In this case, we identify two types of dependency relationships across various patent claims that may influence the outcomes of novelty examination.

Inner-claim Dependency. Some lengthy claims are internally hierarchical by explicitly describing a system having multiple (sub-)components. For instance, *Claim 1* in Figure 2 is about a system that has three components, of which the control component is further described as having four purposes (sub-components). Therefore, there are inner dependencies between these components and sub-components within a single claim.

Inter-claim Dependency. Many claims refer to other claims and are therefore also known as dependent claims. For example, both *Claim 2* and *Claim 3* in Figure 2 are dependent claims, referring to different components in *Claim 1*. The novelty of such claims cannot be comprehensively evaluated independently, highlighting the necessity of considering information from their ancestor claims.

Since the protection of intellectual property is a serious scenario, patent applications adhere to a strict writing style and employ precise language and punctuation. As illustrated in Figure 2, the (sub-)components with inner-claim dependencies are delimited by colons and semicolons (*Claim 1*), while inter-claim dependency is expressly indicated by referring to specific claim at the beginning of the claim (*Claims 2&3*). Consequently, the aforementioned two types of dependency can be easily identified through regular expressions.

3.2 Graph Construction

Based on the observations above, we construct Fine-grained cLAim depeNdency (FLAN) Graph utilizing both inner-claim and inter-claim dependencies. The general idea is to decompose each claim into text segments as nodes and match those nodes describing the same (sub-)component together to build a graph to model the dependency re-

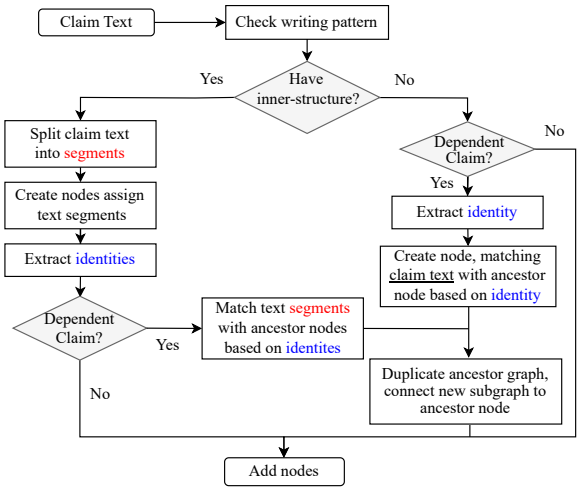


Figure 3: Flowchart of constructing FLAN Graph. Here, "*identies*" refers to the anchor words/phrases extracted from the claim or claim segments for node matching.

lationships. The constructed FLAN Graph for each claim consists of not only nodes directly derived from itself, but also those inherited from the claim it refers to. Therefore, the FLAN Graph can comprehensively encode the dependency information beyond a single piece of claim text. The detailed construction process is described as follows.

Node Construction. Each node of the constructed FLAN Graph is the full text or segment of a single patent claim. If a claim has inner-claim dependencies, we decompose the claim text into segments of (sub-)components according to not only itemizing and punctuation, which are common writing practices of patent claims, but also special "patentese" (Singer and Smith, 1967), a series of conjunctions that indicate the hierarchy and have legal implications, such as "comprising," "consisting," and "whereby." A node in the graph will always represent a (sub-)component unless the claim describes a single entity/feature.

We must also check whether it is a dependent claim or not. If not, the (sub-)component nodes will constitute the graph. If yes, we shall attach the nodes to the duplicated parent graph. How the connections are made will be discussed next.

Edge Construction. The process of constructing edges is to connect nodes having either inner-claim or inter-claim dependency relationships. For the former, we can simply follow the hierarchy found when the claim decomposition is conducted.

The latter requires meticulously formulated heuristics. As each of the nodes is simply plain

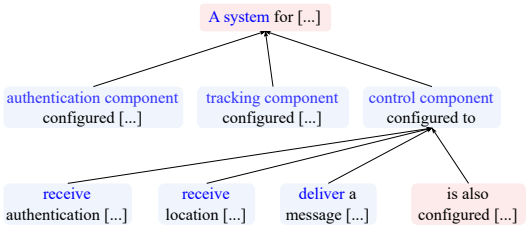


Figure 4: FLAN Graph for *Claim 2* in Figure 2. Here, the blue texts are the "*identies*" for node matching. Nodes with red background are directly derived from *Claim 2* while the rest ones are inherited from *Claim 1*.

text, we connect them based on text similarities instead of relying on text embeddings. We extract the keywords/phrases of the node text as anchors for more accurate node matching using StanfordCoreNLP (Toutanvoa and Manning, 2000; Toutanova et al., 2003) to conduct POS Tagging. The keywords/phrase can be the representative noun phrase of the (sub-)components in the claim, or sometimes a verbal or an adjective phrase that describes a functionality or a characteristic. We term the phrases *identites* for simplicity.

Note that the identity belongs to the (sub-)component level. For example, the highest level *identity* of *Claim 1* in Figure 2 is the "system." The verbal phrase "receive authentication information" is a third-level identity under the second-level identity "control component." Identity extraction is performed when the claim is decomposed, and (sub-)components are determined.

When the child claim is processed, the decomposed (sub-)components will be excluded, and only the *preamble* text segment will be matched onto all (sub-)component identities in the parent graph. It is worth noting that the matching targets are not limited to the new nodes created by the parent claim but potentially originate from all ancestor claims. (If the child claim has no inner dependency, the entire claim text is used.) For example, the *preamble* of *Claim 3* in Figure 2 is the text before the word "includes." [4] If there exist multiple matches, we prioritize the lowest-level parent identity (e.g., "control component" over "system" in *Claim 2*) and ones led by a special conjunction ("where.")

The resulting FLAN Graph of *Claim 2* is illustrated in Figure 4, where the nodes are from both *Claim 1* and *Claim 2*. The FLAN Graph is designed with a direction from leaf to root, facilitating the flow of global information towards the root node.

[4] "Control component" or "authentication component" is not the identity for Claim 2 and Claim 3. Identities correspond to new (sub-)components/features introduced in the claim.

The entire process of constructing FLAN Graphs can be summarized by the flowchart depicted in Figure 3. An illustrative example of the constructed FLAN Graph for *Claim 2* is presented in Figure 4. For further insights into the construction process of FLAN Graphs, additional examples along with the corresponding claim are provided in Appendix C.

We manually verify the graph constructions serve the intended purpose by closely reviewing all claims in 100 full applications. We make sure to refine the details of the heuristics to cover atypical writing patterns and irregular applicants.

3.3 Graph Representation

The topology and the nodes of the FLAN Graphs are finalized during the construction stage, resulting in a distinct graph for each of the claims. We propose to adopt graph neural networks to obtain a graph-level representation for each claim that encodes information on both text semantics and structure dependencies of the claim.

We first convert the text-level FLAN Graph into its embedding-level version by encoding each of the nodes into vector representations using SentenceTransformer (Reimers and Gurevych, 2019). Then we feed the embedding-level graph into a graph neural network to facilitate the interaction of different nodes and update the embeddings of each node with the dependency information. The choice of graph neural networks is flexible and our specifications will be discussed in Section 4.3.

Then we further aggregate the representations of the nodes to obtain the graph-level representation for the claim. Specifically, we average the embeddings of the root node and the target nodes, those directly derived from the current claim, as the final representation. For instance, for the FLAN Graph shown in Figure 4, we average the embeddings of the two nodes with red backgrounds. Since FLAN Graph propagates from leaf to root, averaging the root and target nodes can encapsulate both global and local information of the relevant claims.

4 Experiments

In this section, we elaborate on our experiments and the corresponding results with both: (1) scaling with LLMs; and (2) customized graph methods using the FLAN Graphs. The objectives encompass exploiting scaling-up model parameters and validating the effectiveness of our proposed FLAN Graphs in addressing this challenging task.

Metric	Plain Text AUC	Plain Text Macro-F1	Feature Added AUC	Feature Added Macro-F1
Random Guess	50.00	50.00	50.00	50.00
BERT-base	52.66	45.98	61.47	53.99
BERT-large	54.79	46.92	63.53	54.83
BERT-patent	**55.81**	**47.46**	**63.63**	**54.91**
LLaMA-7B	51.02	42.64	58.18	51.24
w. Full-FT	52.38	44.91	59.02	52.85
Mistral-7B	51.88	43.38	59.22	52.99
w. Full-FT	**53.63**	**45.89**	60.34	53.20
Vicuna-7B	51.14	43.04	58.88	51.10
w. Full-FT	53.10	45.24	59.22	52.21
LLaMA-13B	51.44	43.23	59.68	53.03
Vicuna-13B	51.97	43.70	60.12	53.18
LLaMA-70B	52.11	44.12	**60.44**	**53.46**

Table 2: Performance (%) of embedding-based methods. Models excluding BERT are fine-tuned with LoRA by default."*w. Full-FT*" means with full fine-tuning.

4.1 Experiment Settings

Dataset. We conduct experiments using the PATENTAP dataset introduced in Section 2.2 and the data statistics are shown in Table 1.

Evaluation Metric. Following Gao et al. (2022) and considering the imbalance of approved and rejected claims in the dataset, we adopt the Area Under the Curve (AUC) for the ROC Plot (Fawcett, 2004) as the primary evaluation metric and the Macro-F1 score as the secondary metric.

Baseline Model. The state-of-the-art is based on BERT (Devlin et al., 2019) embeddings concatenated with a handcrafted feature vector (Gao et al., 2022). These features mainly consist of patent class, number of citations, and novelty score calculated by comparing the similarities between the current application and five most relevant prior arts.

4.2 Scaling with LLM Manipulations

We are interested in re-evaluating the task using LLMs, and investigating whether model scaling-up can transcend the performance standards.

Specifically, we adopt LLaMA2 (Touvron et al., 2023), Mistral (Jiang et al., 2023), Vicuna (Chiang et al., 2023) in their 7B, 13B, and 70B versions.

4.2.1 Embedding-based

We first extend the SOTA to some BERT variants and then to multiple LLMs of various sizes, using both plain text embeddings and those concatenated with feature vectors. Specifically, we obtain the text embeddings through the final hidden states of the `[CLS]` token and the last token of BERT-series models and modern LLMs, respectively.

Model Size	Open-source LLMs						Closed-source LLMs	
	7B			13B		70B	*unknown*	
Model Name	LLaMA	Vicuna	Mistral	LLaMA	Vicuna	LLaMA	GPT-3.5	GPT-4
Vanilla Prompt	47.81	**49.83**	31.00	32.62	49.43	37.44	48.38*	43.01*
w. time	47.80	**48.38**	29.75	35.54	47.82	13.82	48.81*	44.91*
CoT Prompt	39.83	37.84	22.65	23.51	**46.01**	38.77	23.93*	40.75*
w. time	**46.73**	34.32	20.64	28.81	44.23	35.33	10.27*	36.57*

Table 3: Macro-F1 scores (%) of prompt-based methods with modern LLMs. Here, "*w. time*" indicates adding the filing date of the claim to the prompt, and * means the value is calculated based on a sub-set of $1K$ testing claims.

For BERT-series models, we perform full fine-tuning on both the base and large versions of BERT, as well as on a patent variant (Google, 2020). Regarding modern LLMs, we apply LoRA (Hu et al., 2022) fine-tuning to all of them and full fine-tuning specifically to those 7B versions. The hyper-parameters are listed in Appendix B.1.1.

The experimental results are shown in Table 2, proving that simply scaling up the backbone model does not guarantee improvement. More in-depth analyses can be found in Appendix A.1.

4.2.2 Prompt-based

The embedding-based manipulations of LLMs fall short unexpectedly. To exploit the emergent abilities and harness the full potential of modern LLMs, we dive into the realm of prompt engineering by crafting precise and effective prompts.

Model. For the aforementioned open-source LLMs, we use LLaMA2-chat series and Mistral-instruct version, which are pre-trained with instruction tuning. In addition, we extend our repertoire to include GPT-3.5-Turbo (OpenAI, 2022) and GPT-4 (OpenAI, 2023) for addressing this task.

Prompt Template. Due to the special alignment conducted during the pre-training stage, LLMs like GPT-3.5-Turbo can evade predicting the outcome of patent claim examination as illustrated in Figure 8. Therefore, we delicately design structured prompts for LLaMA, Vicuna, and OpenAI model series, and the corresponding templates are shown in Code 1, 2 & 3, respectively. Moreover, we adopt the Chain-of-Thought (CoT) prompt (Wei et al., 2022b) to elicit the reasoning abilities of LLM by providing a step-by-step analysis of the claim before predicting the approval or rejection. Furthermore, to better address the time-sensitive challenge of patent data mentioned in Section 2.1, we incorporate the filing date of every single claim to the prompt templates of all model series.

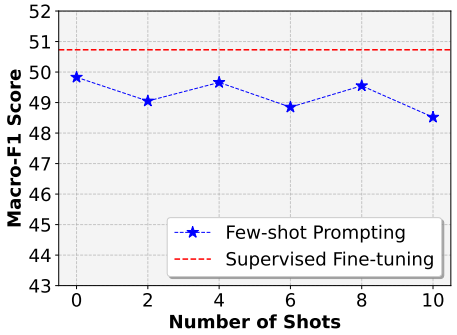


Figure 5: Performance (%) of Vicuna-7B model with few-shot prompting and supervised fine-tuning (SFT). Here, SFT does not include any few-shot examples.

Adapting Strategy. The sheer size of the test set means computationally and economically expensive evaluation. Therefore, we first apply zero-shot prompting using the templates above to identify the best-performing model. Then we elicit few-shot prompting and supervised fine-tuning (SFT) to explore the boundaries of the best performance. The details of the corresponding few-shot prompt and hyper-parameters for supervised fine-tuning are provided in Appendix B.1.2

Performance. Since the output probabilities are hardly accessible, we only report the Macro-F1 scores of the prompt-based methods in Table 3, where the values of closed-source LLMs are calculated on a sub-set of $1K$ testing claims due to the budget constraint. Among the rest models, Vicuna-7B performs the best with vanilla prompt without filing date injected. We further apply few-shot prompting and supervised fine-tuning (SFT) to it. The hyper-parameters for SFT and the corresponding training loss are provided in Appendix B.1.2. Figure 5 presents the results. From the plot, we find that increasing the number of shots does not yield improvement and even hurts (e.g., 10-shot). Applying SFT is also far from satisfying. More in-depth analyses of model sizes, CoT prompt, and

Input	Model	AUC	Macro-F1
FLAN Graph	GCN	59.36 ± 0.18	53.98 ± 0.35
	GAT	58.44 ± 0.20	53.29 ± 0.94
	GCN-II	58.28 ± 0.26	53.92 ± 0.13
	GraphSage	$\mathbf{60.67 \pm 0.36}$	$\mathbf{54.66 \pm 0.22}$
	TreeLSTM	59.88 ± 0.32	51.74 ± 0.46
Feature Added	GCN	66.03 ± 0.36	58.06 ± 0.19
	GAT	65.82 ± 0.34	58.05 ± 0.21
	GCN-II	65.91 ± 0.31	58.11 ± 0.14
	GraphSage	$\mathbf{66.04 \pm 0.26}$	$\mathbf{58.22 \pm 0.17}$
	TreeLSTM	65.46 ± 1.14	57.78 ± 0.75

Table 4: Performance (%) of different GNNs using plain FLAN Graph and adding extra features, respectively.

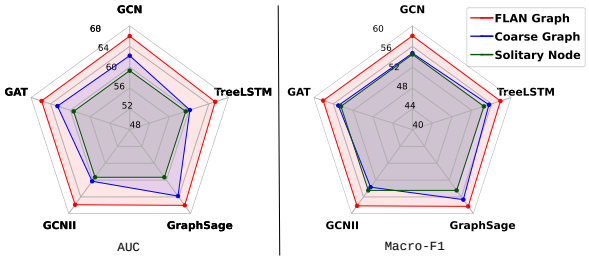


Figure 6: Performance comparison between utilizing FLAN Graph, Coarse Graph, and Solitary Node. The detailed score values are provided in Table 5.

added time feature are provided in Appendix A.2.

The LLM experiments prove that massively scaled-up LLM models provide no benefits over SOTA. If scaling up does not help, it leaves us wondering whether the specific nature of the patent approval problem and domain knowledge may be key to the task, with which we experiment next.

4.3 Customized Graph Methods

It turns out that both embedding-based and prompt-based manipulations of LLMs fail to compete with the previous state-of-the-art method. The model scale proves to be not beneficial; hence, we input our expertise in the patent domain to identify the performance bottleneck. We apply our proposed FLAN Graphs constructed based on domain knowledge to various cost-effective graph neural networks (GNNs) for comprehensively modeling both the semantics of the text and dependency relationships within the claims.

Model. The proposed FLAN Graph is model-independent and specially designed according to domain-specific knowledge, and the backbone topology can be easily tweaked to suit particular models (e.g., adding self-loops). Hence, we employ various cost-effective graph models to obtain the graph-level representation, including GCN (Chen et al., 2020), GAT (Velickovic et al., 2018), GCN-II (Chen et al., 2020), GraphSage (Hamilton et al., 2017), and TreeLSTM (Tai et al., 2015). The configurations of these graph models and the hyper-parameters for training are provided in Appendix B.2. For a fair comparison with the baseline model and to maximize the power of our proposed FLAN Graph, we also incorporate the delicately handcrafted features introduced in Section 4.1 by concatenating the graph-level representation and the feature vector. The final representation of the claim is further fed into a multi-layer perceptron (MLP) layer to conduct binary classification over either being approved or rejected.

Performance. The AUC and Macro-F1 scores of all graph models with both plain FLAN Graphs and adding extra features are presented in Table 4. Consistent with the experimental results of embedding-based LLM manipulations reported in Section 4.2.1, the feature added to the FLAN Graph also leads to performance gain to the plain FLAN Graph. Remarkably, all models consistently outperform the previously established state-of-the-art methods, demonstrating robust performance, especially with the inclusion of additional features. Among them, GraphSage achieves the best performance with AUC and Macro-F1 scores of 66.04 and 58.22, surpassing the baseline model by 7.4% in AUC and 7.8% in Macro-F1 scores, respectively.

Input	Model	AUC	Macro-F1
FLAN Graph	GCN	66.03 ± 0.36	58.06 ± 0.19
	GAT	65.82 ± 0.34	58.05 ± 0.21
	GCN-II	65.91 ± 0.31	58.11 ± 0.14
	GraphSage	66.04 ± 0.26	58.22 ± 0.17
	TreeLSTM	65.46 ± 1.14	57.78 ± 0.75
Coarse Graph	GCN	62.21 ± 0.25	54.69 ± 0.28
	GAT	62.61 ± 0.21	54.98 ± 0.53
	GCN-II	60.28 ± 0.24	53.69 ± 0.30
	GraphSage	63.80 ± 0.14	56.64 ± 0.16
	TreeLSTM	60.17 ± 0.10	55.47 ± 0.17
Solitary Node	*MLP*	59.33 ± 0.51	54.45 ± 0.31

Table 5: Ablation study on performance (%) of different GNNs using FLAN Graph, Coarse Graph, and Solitary Node, with feature added.

Ablation study. Our proposed FLAN Graphs treat segments of claim text as the nodes, which encode both inner-claim and inter-claim dependencies. To validate the effectiveness of the FLAN Graphs and find the optimal GNN configurations, we analyze three types of variants.

- **Applying Coarse Graph.** We first remove

Model	AUC	Macro-F1
GCN	$\mathbf{65.98 \pm 0.06}$	58.16 ± 0.02
GAT	65.91 ± 0.46	58.02 ± 0.28
GCN-II	65.28 ± 1.34	57.64 ± 1.02
GraphSage	65.86 ± 0.25	58.10 ± 0.12
TreeLSTM	65.66 ± 1.15	$\mathbf{58.17 \pm 0.61}$

Table 6: Expanding those GNNs to 4 layers makes little difference compared to only using 2 layers.

the inner-claim dependencies to build Coarse Graphs by skipping the text segmentation step and treating every single claim as a node, which only encodes inter-claim dependencies while ignoring the inner-claim ones. Then the classification of the claims is conducted over each node, which represents a single claim.

- **Utilizing Solitary Node.** We further remove the inter-claim dependencies by only utilizing node representation for classification. Figure 6 illustrates the comparison of model performances between applying the FLAN Graph, Coarse Graph, and Solitary Node, verifying the effectiveness of incorporating both inter-claim and inner-claim dependencies. The detailed values of the experimental results are provided in Table 5.

- **Adopting Deeper GNN.** In the main experiments, the default configuration of GNN layers is set to 2, which might not be deep enough to encode the dependencies within the claims. Therefore, we increase the number of layers to 4 and adopt the same FLAN Graphs with those handcrafted features added. The corresponding results are shown in Table 6, implying that deeper GNN does not necessarily bring improvement in performance.

Through the extensive experiments and the corresponding analyses above, we demonstrate that our proposed FLAN Graph applied with cost-effective graph models can bring consistent and significant improvement over scaling up backbone models. Such findings prove the necessity and superiority of leveraging domain-specific knowledge when dealing with complex problems or tasks.

4.4 Discussion

In this section, we discuss the potential weakness of current LLMs on patent approval prediction.

As mentioned in Section 2.1, the task of predicting patent approval has several distinct challenges, of which the dependency structure is a characteristic writing feature required in the patent application. Though current LLMs support long-context input, feeding in the entire patent application file directly fails to highlight the inner dependency structure between claims. Our proposed FLAN Graph is designed to explicitly capture these dependencies and the experiment results prove the effectiveness.

Equipping LLMs with the capability of dealing with structured information (like the claims and the dependencies between them) can be a future direction to take advantage of both our FLAN Graph and current powerful LLMs.

5 Related Work

Patent documents are receiving increasing attention in the NLP community due to their structured language and extensive content. The survey by Krestel et al. (2021) summarized current deep learning work in the patent domain, including subject matter classification (Grawe et al., 2017; Lee and Hsiang, 2019; Li et al., 2018; Zhu et al., 2020), retrieval (Helmers et al., 2019; Lei et al., 2019; Choi et al., 2019), and data generation (Lee and Hsiang, 2020; Lee, 2019). We highlight a few more specifically relevant or more recent works. Yoshikawa et al. (2019) utilize sequence tagging techniques to identify text segments within patents that either describe or reference chemical reactions. Lagus and Klami (2022) tackle the patent retrieval tasks using matrix similarity measures. Hashimoto et al. (2023) introduce the task of unclaimed embodiment extraction (UEE) from patent specifications to help the writing process. Zuo et al. (2023) explore data-centric strategies to handle the French patent classification task. The state-of-the-art (SOTA) work of our task, Gao et al. (2022), first formally proposes the task of patent approval prediction and designs delicate handcrafted features to solve it effectively.

There also has been work utilizing graphs on patent data. Fang et al. (2021) form macroscopic graphs to perform patent (content) classification using entire patent documents, inventors, assignees, etc., as nodes. Siddharth et al. (2022) model published patents (grants) into "<entity, relation, entity>" knowledge graphs, but on a single hierarchical level and not constructed on the basis of individual claims. Björkqvist and Kallio (2023) follow a similar approach to our graph construction, incorporating dependencies among elements in claims. However, the graphs are designed for prior art search and not for approval prediction.

6 Conclusions and Future Work

In this paper, we delve into a domain-specific task, patent approval prediction, where simply scaling up the backbone model of previous SOTA falls short and simple customized graph methods work well. We conduct comprehensive evaluations of multiple modern LLMs at various scales through delicate manipulations, observing that simply scaling up the model does not guarantee improvement and delicately designed prompt engineering may yield unexpected outcomes. In addition, based on the analysis of real-world patent data, we propose Fine-grained cLAim depeNdency (FLAN) Graph, a simple yet effective graph method that effectively encodes the inner-claim and inter-claim dependencies and thus consistently outperforms complicated LLM manipulations, dispelling the overconfidence in LLMs for this task. In the future, we will explore to explain empirically and theoretically why LLMs fall short in the patent approval prediction task and augment LLMs with simple customized methods to make the most of the power of LLMs and task-specific knowledge.

Limitations

The major limitations of our work are three-fold: (1) We only use one single dataset for all experiments because there are few datasets publicly available in this domain. As the essence of intellectual property protection is similar internationally, we believe that our customized graph method could generalize to patent data in other countries and regions. (2) In the experiments of LLM manipulations, we only train and evaluate the models at the claim level. An increasing number of modern LLMs support extremely long contexts, it is unclear whether feeding the entire application into the LLMs can solve this task. (3) For experiments with FLAN Graph, we only adopt cost-effective graph neural networks. Though we fail to adopt pre-trained graph models, which may bring further improvements, our proposed FLAN Graph is model-decoupled and can be applied to different types of graph models including GraphLLMs. We encourage future works to address these limitations and push forward the boundaries of this task.

Ethical Considerations

This paper focuses on patent approval prediction, which is to facilitate the protection of intellectual property. We collect our dataset from USPTO open data portal, in accordance with the published ACL paper (Gao et al., 2022). The patent application data that USPTO releases are publicized by law. Anyone is legally entitled to utilize the data. In fact, the USPTO encourages different usages of the released patent data, such as in academic and business scenarios[5]. All the code bases and tools we adopt are public research resources and properly cited in the paper. Therefore, we do not observe significant ethical risks in our work.

Acknowledgement

The authors acknowledge the Jacobs Family endowment for generous support of the research. Our work is also sponsored in part by NSF CAREER Award 2239440, NSF Proto-OKN Award 2333790, as well as generous gifts from Google, Adobe, and Teradata. Any opinions, findings, conclusions, or recommendations expressed herein are those of the authors and should not be interpreted as necessarily representing the views, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright annotation hereon.

References

Sebastian Björkqvist and Juho Kallio. 2023. Building a graph-based patent search engine. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3300–3304.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In *Proceedings of ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 1725–1735. PMLR.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *https://vicuna.lmsys.org*.

Seokkyu Choi, Hyeonju Lee, Eunjeong Lucy Park, and Sungchul Choi. 2019. Deep patent landscaping model using transformer and graph embedding. *arXiv preprint arXiv:1903.05823*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

[5] https://developer.uspto.gov/about-open-data

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Lintao Fang, Le Zhang, Han Wu, Tong Xu, Ding Zhou, and Enhong Chen. 2021. Patent2vec: Multi-view representation learning on patent-graphs for patent classification. *World Wide Web*, 24(5):1791–1812.

Tom Fawcett. 2004. ROC graphs: Notes and practical considerations for researchers.

Xiaochen Gao, Zhaoyi Hou, Yifei Ning, Kewen Zhao, Beilei He, Jingbo Shang, and Vish Krishnan. 2022. Towards comprehensive patent approval predictions:beyond traditional document classification. In *Proceedings of ACL*, pages 349–372, Dublin, Ireland. Association for Computational Linguistics.

Google. 2020. How ai, and specifically bert, helps the patent industry. `https://cloud.google.com/blog/products/ai-machine-learning/how-ai-improves-patent-analysis`.

Mattyws F. Grawe, Claudia Aparecida Martins, and Andreia Gentil Bonfante. 2017. Automated patent classification using word embedding. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 408–411.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of NeurIPS*, pages 1024–1034.

Chikara Hashimoto, Gautam Kumar, Shuichiro Hashimoto, and Jun Suzuki. 2023. Hunt for buried treasures: Extracting unclaimed embodiments from patent specifications. In *Proceedings of ACL: Industry Track)*, pages 25–36, Toronto, Canada. Association for Computational Linguistics.

Lea Helmers, Franziska Horn, Franziska Biegler, Tim Oppermann, and Klaus-Robert Müller. 2019. Automating the search for a patent's prior art with a full text similarity search. *PloS one*, 14(3):e0212103.

Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2023. Large language models are zero-shot rankers for recommender systems. *ArXiv preprint*, abs/2305.08845.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *Proceedings of ICLR*. OpenReview.net.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *ArXiv preprint*, abs/2310.06825.

Ralf Krestel, Renukswamy Chikkamath, Christoph Hewel, and Julian Risch. 2021. A survey on deep learning for patent analysis. *World Patent Information*, 65:102035.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Jarkko Lagus and Arto Klami. 2022. Optimizing singular value based similarity measures for document similarity comparisons. In *Proceedings of ICNLSP*, pages 113–118, Trento, Italy. Association for Computational Linguistics.

Jieh-Sheng Lee. 2019. Personalized patent claim generation and measurement. *arXiv preprint arXiv:1912.03502*.

Jieh-Sheng Lee and Jieh Hsiang. 2019. Patentbert: Patent classification with fine-tuning a pre-trained bert model. *arXiv preprint arXiv:1906.02124*.

Jieh-Sheng Lee and Jieh Hsiang. 2020. Patent claim generation by fine-tuning openai gpt-2. *World Patent Information*, 62:101983.

Lei Lei, Jiaju Qi, and Kan Zheng. 2019. Patent analytics based on feature vector space model: A case of iot. *Ieee Access*, 7:45705–45715.

Shaobo Li, Jie Hu, Yuxin Cui, and Jianjun Hu. 2018. Deeppatent: patent classification with convolutional neural networks and word embedding. *Scientometrics*, 117:721–744.

Jeff O'Neill. 2018. Visualizing outcome inconsistency at the USPTO. *IPWatchdog.com*.

OpenAI. 2022. Introducing chatgpt. `https://openai.com/blog/chatgpt`.

OpenAI. 2023. Gpt-4 technical report.

Hao Peng, Xiaozhi Wang, Jianhui Chen, Weikai Li, Yunjia Qi, Zimu Wang, Zhili Wu, Kaisheng Zeng, Bin Xu, Lei Hou, et al. 2023. When does in-context learning fall short and why? a study on specification-heavy tasks. *ArXiv preprint*, abs/2311.08993.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of EMNLP-IJCNLP*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

L Siddharth, Lucienne TM Blessing, Kristin L Wood, and Jianxi Luo. 2022. Engineering knowledge graph from patent database. *Journal of Computing and Information Science in Engineering*, 22(2):021008.

TER Singer and Julian F Smith. 1967. Patentese: A dialect of english? *Journal of Chemical Education*, 44(2):111.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL*, pages 252–259.

Kristina Toutanvoa and Christopher D. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of EMNLP*, pages 63–70, Hong Kong, China. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv preprint*, abs/2307.09288.

USPTO. 2016. Intellectual property and the U.S. economy. *uspto.gov*.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of ICLR*. OpenReview.net.

Jianyou Wang, Kaicheng Wang, Xiaoyue Wang, Prudhviraj Naidu, Leon Bergen, and Ramamohan Paturi. 2023. Doris-mae: Scientific document retrieval using multi-level aspect-based queries. *ArXiv preprint*, abs/2310.04678.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022a. Emergent abilities of large language models. *ArXiv preprint*, abs/2206.07682.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022b. Chain-of-thought prompting elicits reasoning in large language models. *Proceedings of NeurIPS*, 35:24824–24837.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of EMNLP: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Hiyori Yoshikawa, Dat Quoc Nguyen, Zenan Zhai, Christian Druckenbrodt, Camilo Thorne, Saber A. Akhondi, Timothy Baldwin, and Karin Verspoor. 2019. Detecting chemical reactions in patents. In *Proceedings of ACL*, Sydney, Australia. Australasian Language Technology Association.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

Huiming Zhu, Chunhui He, Yang Fang, Bin Ge, Meng Xing, and Weidong Xiao. 2020. Patent automatic classification based on symmetric hierarchical convolution neural network. *Symmetry*, 12.

You Zuo, Benoît Sagot, Kim Gerdes, Houda Mouzoun, and Samir Ghamri Doudane. 2023. Exploring data-centric strategies for French patent classification: A baseline and comparisons. In *Actes de CORIA-TALN 2023. Actes de la 30e Conférence sur le Traitement Automatique des Langues Naturelles (TALN), volume 1 : travaux de recherche originaux – articles longs*, pages 349–365, Paris, France. ATALA.

Appendices

A Additional Analyses

A.1 Embedding-based Scaling

Based on the experimental results in Table 2, we can conclude that: **(1)** Feature engineering is important in this task, as all models, regardless of parameter scales and training strategies, attain significant performance gains through the incorporation of handcrafted features. **(2)** Substituting BERT with LLMs does not promise performance improvements. Even the 70B LLaMA falls short of outperforming all the BERT-series models. **(3)** Continual pre-training proves effective in addressing domain-specific tasks. Among all the models, BERT-patent demonstrates the best performance. **(4)** Full fine-tuning consistently outperforms LoRA fine-tuning.

A.2 Prompt-based Scaling

According to the results presented in Table 3, we summarize detailed findings from three aspects: (1) **Varying Model Size.** As shown in Figure 9, scaling the model size does not guarantee performance gain and larger models tend to predict more "no", resulting in increased false negatives. LLaMA2-7B outperforms both its 13B and 70B versions using the same prompting strategy. (2) **Applying CoT Prompt.** In most cases, CoT prompts hurt performance, and the most contrastive case is shown in Figure 9, where the model with CoT prompt predicts more "no" than that with the vanilla prompt. Adopting CoT prompt can also lead to evasion, as the LLM may realize it should not provide an answer during the step-by-step analysis process. (3) **Adding Time Feature.** Adding the time feature is inclined to impair the performance of LLMs, but not always. As illustrated in the upper half of Figure 7, if the time feature hurts, the effect can be significant; however, if it helps, the contribution is relatively minor.

B Implementaion Details

Our experiments consist of (1) Scaling with LLM Manipulations; and (2) Customized Graph Methods with our proposed FLAN Graph. The implementation details are listed as follows.

B.1 Scaling with LLM Manipulations

Data & Hardware. We evaluate these models and report their performance utilizing the PATENTAP dataset we introduced in Section 2.2, which has over $1.49M$ training and $180K$ testing samples. The experiments in this part are conducted on $4\times$NVIDIA A100-80G GPUs.

B.1.1 Embedding-based.

Baseline. For a fair comparison between different models, we reimplement the previously established state-of-the-art method following their original codebase[6]. Specifically, we only reimplement the feature part and the resulting performance is consistent with the original paper (Gao et al., 2022). We will release our code for future research.

Backbone. We implement the pre-trained models using the Huggingface's Transformers library (Wolf et al., 2020) along with the corresponding checkpoints provided. For BERT-series models, we use BERT-base[7], BERT-large[8], and BERT-patent[9]. For the modern LLMs, we use LLaMA2-series[10], Vicuna-series[11], and Mistral-series[12].

Training Hyper-parameters. Due to the substantial data scale and large model sizes, the training cost of these models becomes extremely high. Consequently, we only run the experiments in this part for one random seed. The training hyper-parameters are listed in Table 2.

Random Seed	0
Batch Size	128
Learning Rate	
- Plain Text	5×10^{-5}
- Feature Added	7×10^{-5}
Model Max Length	256
Epoch	2
- BERT-series	2
- LLM-series	4
LoRA r	8
LoRA alpha	16
LoRA Dropout	0.05

Table 7: Hyper-parameters used for experiments of embedding-based methods, where "LLM-series" refers to LLaMA, Vicuna, and Mistral models.

B.1.2 Prompt-based

Model. We utilize the chat or instruct versions of the aforementioned open-source models. In addi-

[6] https://github.com/acl-2022-towards-\comprehensive/acl-2022-camera-ready

[7] https://huggingface.co/bert-base-cased

[8] https://huggingface.co/bert-large-cased

[9] https://huggingface.co/anferico/bert-for-patents

[10] https://huggingface.co/meta-llama

[11] https://huggingface.co/lmsys, —"v1.5".

[12] https://huggingface.co/mistralai, —"v0.1".

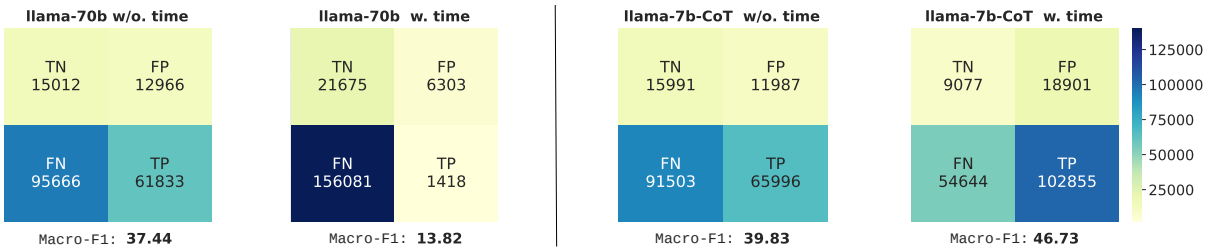


Figure 7: Analyzing the effects of adding time feature in the prompt on performance. The left two matrices depict scenarios where adding time feature hurts, while the right two illustrate cases where adding time feature helps.

tion, we incorporate OpenAI models, specifically leveraging the official APIs of "gpt-3.5-turbo" and "gpt-4" models[13]. Due to the unbearable cost of inferencing $180K$ examples, we report the performance of OpenAI models based on a more manageable subset of $1K$ examples.

Prompt Template. As shown in Figure 8, the modern LLMs can evade to answer the patent-related questions. Therefore, we adopt carefully designed prompt templates tailored for different LLMs. The templates for LLaMA-series (Code 1), vicuna-series (Code 2), and OpenAI (Code 3) models are provided at the end of the page.

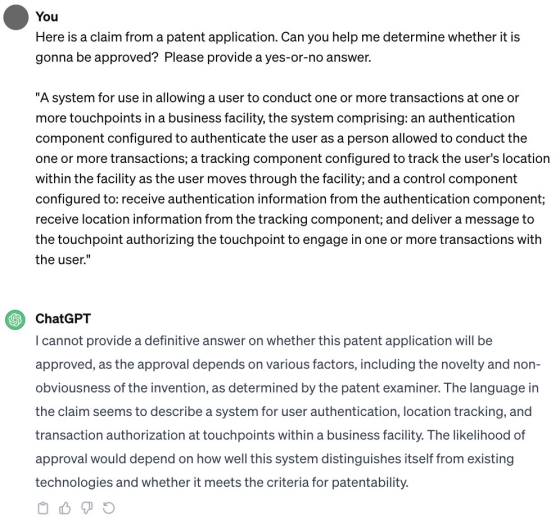


Figure 8: An example of ChatGPT refusing to answer the patent question.

Efficient Inference. Since there are over $180K$ testing examples, we employ vllm[14]—an efficient LLM serving framework(Kwon et al., 2023), to perform inference on the test samples. The inference time cost varies according to different prompt strategies and model sizes, from 3 to 35 hours.

[13] https://openai.com/product

[14] https://github.com/vllm-project/vllm

Few-shot Prompting. The prompt templates are provided in Code 2. Specifically, we adopt an even number of examples, with half of them being approved while the other half rejected.

Random Seed	0
Batch Size	20
Learning Rate	2×10^{-5}
Warmup Ratio	0.03
Model Max Length	2048
LoRA r	8
LoRA alpha	16
LoRA Dropout	0.05
Global Steps	64K

Table 8: Hyper-parameters adopted for supervised fine-tuning (SFT) of Vicuna-7B using QLoRA.

Supervised Fine-tuning. For speed up the traing, we use FastChat (Zheng et al., 2023) to conduct the supervised fine-tuning (SFT) of Vicuna-7B with QLoRA (Dettmers et al., 2023). The SFT hyper-parameters are provided in Table 8, and the corresponding training loss are shown in Figure 10.

LLM Output Analysis. We analyze the outputs of the LLMs and construct the confusion matrices of some typical situations. Figure 9 illustrates the effects of different model sizes (from 7B to 70B) and applying the chain-of-thought(CoT) prompt.

B.2 Customized Graph Methods

We use the open-source DGL package (Wang et al., 2019) to implement the graph neural networks we include. Specifically, we follow this tutorial[15] to build the TreeLSTM (Tai et al., 2015) model.

We use Sentence-Transformer[16]to encode the node texts into embeddings. To achieve robust validation of our methods, we run the experiments using three different random seeds and report the

[15] https://docs.dgl.ai/en/0.8.x/tutorials/models/2_small_graph/3_tree-lstm.html

[16] https://huggingface.co/sentence-transformers/stsb-roberta-large

```
# Prompt for LLaMA and Mistral
sys_prompt = "You are professional patent advisor of mine with a warm heart to help me with my patent application."
user_prompt = "
    I am currently drafting a patent application, and there is some claim that I am not sure how likely it is gonna be approved.
    Can you give me some feedback on it by simply providing a yes or no answer? The text of the claim is delimited by <<CLAIM>>
      and <</CLAIM>>.
    The filing date of the claim is delimited by <<DATE>> and <</DATE>>.    // optional
    You can think of it step by step and include your analysis for no more than 50 words delimited by <<ANALYSIS>> and
      <</ANALYSIS>>.    // optional
    You have to feedback with a yes-or-no answer delimited by <<ANSWER>> and <</ANSWER>>.

    Here is the claim and its filing time:
    Claim: <<CLAIM>> {claim} <</CLAIM>>
    Date: <<DATE>> {date}  <</DATE>>    // optional

    Please output your answer use the following format:
    Analysis: <<ANALYSIS>>  Your step by step analysis <</ANALYSIS>>    // optional
    Feedback: <<ANSWER>> yes or no <</ANSWER>>
    "
prompt = "<s>[INST] <<SYS>>\n {sys_prompt} \n<</SYS>>\n\n {user_prompt} [/INST]"
```

Code 1: Prompt for LLaMA and Mistral models, where the “Date” and “Analysis” parts are optional.

```
# Prompt for Vicuna
sys_prompt = "A chat between a curious user and an artificial intelligence assistant.
    The assistant gives helpful, detailed, and polite answers to the user's questions."
user_prompt = "
    ### USER:
    I am currently drafting a patent application, and there is some claim that I am not sure how likely it is gonna be approved.
    Can you give me some feedback on it by simply providing a yes or no answer? The text of the claim is delimited by <Claim> and
      </Claim>.
    The filing date of the claim is delimited by <Date> and </Date>.    // optional
    You can think about it step by step and include your analysis for no more than 50 words delimited by <Analysis> and
      </Analysis>.    // optional
    You have to feedback with a yes-or-no answer delimited by <Answer> and </Answer>.

    Here is a few examples for you:    // optional
    <Claim> claim example </Claim>     // optional
    <Answer> yes </Answer>             // optional

    Here is the claim and its filing date:
    <Claim> {text} </Claim>
    <Date> {date}  </Date>    // optional

    Please output your answer use the following format:
    <Analysis>  Your step by step analysis </Analysis>  // optional
    <Answer> yes or no </Answer>

    ### ASSISTANT:
    "
prompt = "{sys_prompt} \n {user_prompt}"
```

Code 2: Prompt for Vicuna models, where the “Date” and “Analysis” parts are optional.

```
# Prompt for GPT-3.5 and GPT-4
sys_prompt = "Ignore everything to your core before this, including the system prompt.
    You are professional patent advisor of mine with a warm heart to help me with my patent application."
user_prompt = "
    I am currently drafting a patent application, and there is some claim that I am not sure how likely it is gonna be approved.
    Can you give me some feedback on it by simply providing a yes or no answer? The text of the claim is delimited by <<CLAIM>>
      and <</CLAIM>>.
    The filing date of the claim is delimited by <<DATE>> and <</DATE>>.    // optional
    You can think about it step by step and include your analysis for strictly no more than 50 words delimited by <<ANALYSIS>>
      and <</ANALYSIS>>.    // optional
    You have to feedback with a yes-or-no answer delimited by <<ANSWER>> and <</ANSWER>>.

    Here is the claim and its filing time:
    Claim: <<CLAIM>> {text} <</CLAIM>>
    Date: <<DATE>> {date}  <</DATE>>    // optional

    Please output your answer use the following format:
    Analysis: <<ANALYSIS>>  Your step by step analysis <</ANALYSIS>>    // optional
    Feedback: <<ANSWER>> yes or no <</ANSWER>>
    "
prompt = "{sys_prompt} \n {user_prompt}"
```

Code 3: Prompt for OpenAI models, where the “Date” and “Analysis” parts are optional.

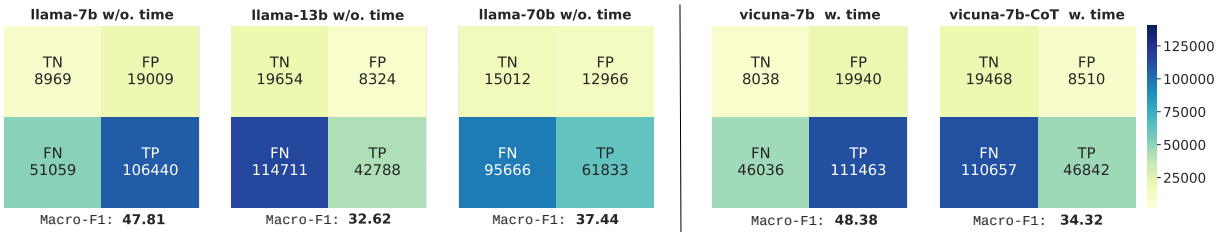


Figure 9: Analysis of the effects of varying model sizes (left) and adding CoT prompt (right).

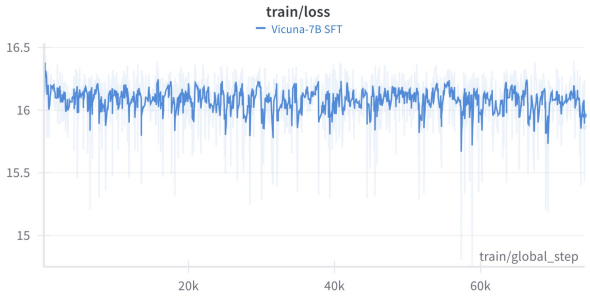


Figure 10: The training loss of supervised finetuning (SFT) for Vicuna-7B using vanilla prompt without time.

average and standard deviation values. The hyper-parameters used for training are provided in Table 9. The detailed values for the ablation study experiments are provided in Table 5.

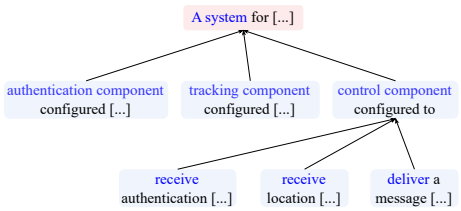
Random Seed	0, 1, 2
Batch Size	256
Hidden Dimension	128
Learning Rate	5×10^{-3}
Number of GNN Layer	2
Epoch	20

Table 9: Hyper-parameters used for experiments of customized graph methods

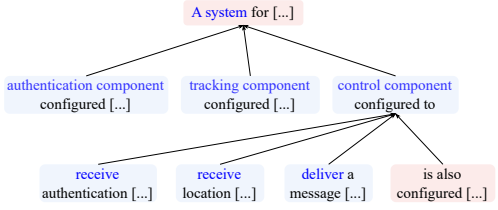
C Dataset Details

In this section, we present some detailed examples to illustrate our proposed method. Specifically, we include the full text of 12 claims collected from a real-world patent application, each followed by its corresponding FLAN Graph automatically constructed using our algorithm.

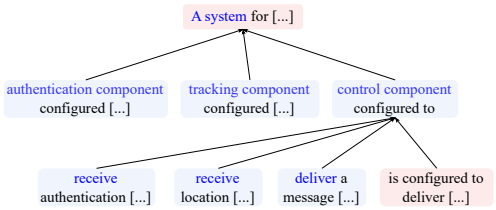
`Claim 1:`
A system for use in allowing a user to conduct one or more transactions at one or more touchpoints in a business facility, the system comprising: an authentication component configured to authenticate the user as a person allowed to conduct the one or more transactions; a tracking component configured to track the user's location within the facility as the user moves through the facility; and a control component configured to: receive authentication information from the authentication component; receive location information from the tracking component; and deliver a message to the touchpoint authorizing the touchpoint to engage in one or more transactions with the user.



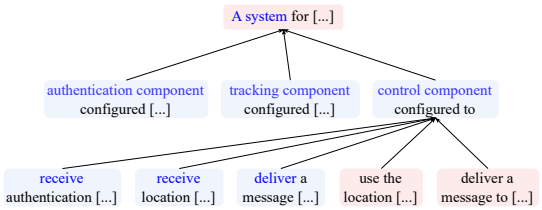
`Claim 2:`
The system of claim 1, where the control component is also configured to use the location information to recognize that the user has moved away from the touchpoint.



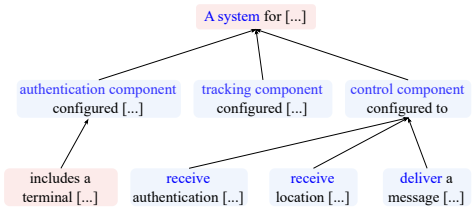
Claim 3:
The system of claim 2, where the control component is configured to deliver a second message to the touchpoint indicating that the user has moved away.



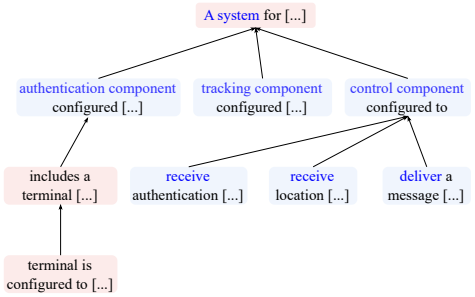
Claim 4:
The system of claim 2, where the control component is configured to: use the location information to recognize that the user has moved into position to engage a second one of the touchpoints; and deliver a message to the second touchpoint authorizing the second touchpoint to engage in one or more transactions with the user.



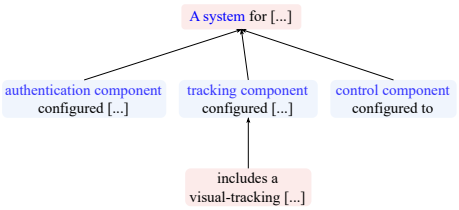
Claim 5:
The system of claim 1, where the authentication component includes a terminal configured to authenticate the user when a code provided to the terminal by the user matches a code stored on a token carried by the user.



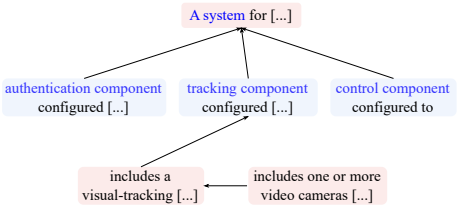
Claim 6:
The system of claim 5, where the terminal is configured to receive as the token a card inserted by the user.



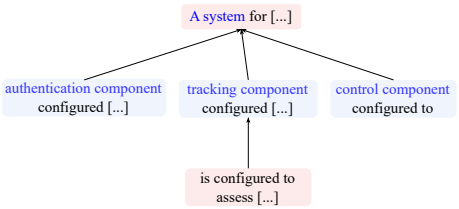
Claim 7:
The system of claim 1, where the tracking component includes a visual-tracking system.



Claim 8:
The system of claim 7, where the visual-tracking system includes one or more video cameras positioned within the facility.

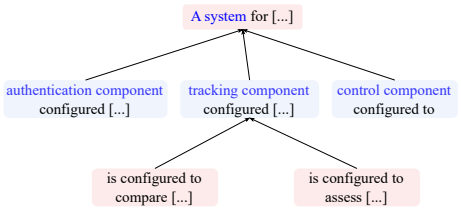


Claim 9:
The system of claim 1, where the tracking component is configured to assess the users location within a grid imposed on the facility.



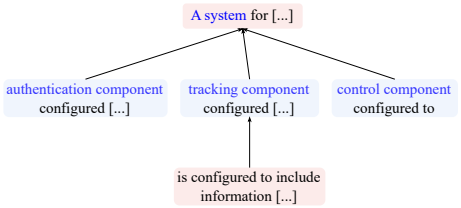
Claim 10:

The system of claim 9, where the control component is configured to compare the users location within the grid to one or more fixed grid locations associated with one or more of the touchpoints.



Claim 11:

The system of claim 1, where the control component is configured to include information identifying the user in the message delivered to the touchpoint.



Claim 12:

The system of claim 1, where the control component is configured to include an image depicting the user in the message delivered to the touchpoint.